UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/772,207	02/03/2004	Sebastian Lange	MS1-1823US	4217

22801 7590 01/06/2009

LEE & HAYES, PLLC
601 W. RIVERSIDE AVENUE
SUITE 1400
SPOKANE, WA 99201

EXAMINER	
DOAN, TRANG T	
ART UNIT	**PAPER NUMBER**
2431	

MAIL DATE	DELIVERY MODE
01/06/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/772,207	LANGE ET AL.	
	Examiner	Art Unit	
	TRANG DOAN	2431	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 October 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9, 11-15, 17-22, 24-34, 36-43, 45 and 51* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9,11-15,17-22,24-34,36-43,45 and 51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02/23/2004* is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 10/22/2008.

2. Claims 1, 12, 14, 19, 26 and 38 have been amended.

3. Claim 51 is new.

4. Claims 10, 16, 23, 35, 44 and 46-50 have been canceled.

5. Claims 1-9, 11-15, 17-22, 24-34, 36-43, 45 and 51 are pending for consideration.

Continued Examination Under 37 CFR 1.114

6. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 10/22/2008 has been entered.

Response to Arguments

7. Applicant's arguments with respect to claims 1-9, 11-15, 17-22, 24-34, 36-43, 45 and 51 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 101

8. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

9. Claims 1-9, 12-15, 17, 19-22 and 24-25 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

10. Claims 1-9, 11-15 and 17 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Applicant's claims are directed to a method for estimating security requirements. Method claims are generally considered processes under § 101. A process is statutory under § 101 only if the process is tied to another statutory class (such as a particular apparatus) or it transforms underlying subject matter to a different state or thing.

11. Applicant's claims 1 and 12 merely recite the two steps for simulating and finding permission sets for each execution path. The claims do not recite any limitation that ties the process to another statutory class, such as a particular machine that accomplishes the process steps, nor does the claim recite any subject matter that is being transformed by the steps. The dependent claims do not remedy these deficiencies. Therefore, claims 1-9, 12-15 and 17 are rejected for being directed towards non-statutory subject matter. This rejection may be overcome by amending the method claim with language that will tie the method to another statutory class.

12. Claims 19-22 and 24-25 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. a) As to claim 19, this claim is directed to a computer-readable storage media having a tangible component comprising instructions. According to the specification, paragraph 0054, "computer readable media may comprise "computer storage media" and "communications media (e.g., carrier waves, infrared signals, digital signals, etc.)". In light of the specification, this claim

does not fall within one of the four statutory classes of an invention (method/process, article of manufacture, a composition of matter, or machine). Carrier wave is a signal, not a series of steps. Carrier wave is a form of energy and not a composition of matter. Carrier wave does not have any physical structure, does not itself perform any useful, concrete and tangible result and thus does not fit within the definition of a machine or an article of manufacture.

13. The dependent claims are depended on the rejected base claim, and are rejected for the same rationales.

Claim Rejections - 35 USC § 112

14. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

15. Claims 1-9, 12-15 and 17 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

16. Regarding claims 1 and 12, Applicant claims two methods in the same claim. The first one is a method to estimate security requirements in line 1 and the other one is a public entry point of a method in the assembly in line 11. Appropriate correction is required.

17. The dependent claims are depended on the rejected base claim, and are rejected for the same rationales.

Claim Rejections - 35 USC § 103

18. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

19. Claims 1-9, 11-15, 17-22, 24-34, 36-43, 45 and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Koved et al. (reference U) (hereinafter Koved) in view of Scheifler et al. (US 2002/0174224) (hereinafter Scheifler).

Regarding claim 1, Koved discloses simulating the execution of all execution paths of one or more assemblies in managed code, wherein an assembly comprises one or more files versioned and deployed as a unit, wherein the managed code is a managed shared library or an executable (Koved: on page 1, column 2, under INTRODUCTION heading, second paragraph: “developer reads … libraries used (including the Java run-time libraries”) and reduces the required access rights), wherein all managed code is contained within the one or more assemblies (Koved: on page 2, column 2); and finding a set of required permissions for each execution path by one or more simulated stack walks that each include a plurality of the assemblies, wherein each call in each execution path has a corresponding said permissions set (Koved: on page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java … at each stack frame, as well as the result of any authorization test encountered” and page 3, column 1

and first paragraph: “the runtime stack … to discover authorization requirements by analyzing all possible paths through the program”).

Koved does not explicitly disclose wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; gathering a permission set for the called method; and creating a union of the gathered permission sets. However, Scheifler discloses wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; gathering a permission set for the called method; and creating a union of the gathered permission sets (Scheifler: paragraphs 0016, 0083-0086, 0089, 0091, 0126, 0130 and 0139-0140). Therefore, it would have been obvious to a person skilled art at the time the invention was made to have included in Koved the feature of Scheifler as discussed above in order to increase security by providing flexible designation of access privileges to code. The systems and methods not only base the access privileges on the source of the code (i.e., whether the code is trusted or untrusted), but also on the identity of the principal on whose behalf the code is being executed (i.e., whether the principal requesting the code execution is trusted or untrusted) (Scheifler: paragraph 0026).

Regarding claim 2, Koved as modified discloses wherein the execution paths for only one said assembly in managed code are simulated to find the set of required permissions for each said execution path by a union of the permissions for each said execution path (Koved: on page 2, column 2, third paragraph; page 3, column 1, first

paragraph; and page 3 and page 4, under Authorization Model section: "In this paper…an invocation graph and data flow analysis…more accurate authorization information." "Our approach…discover authorization requirements by analyzing all possible paths through the program." "It can be seen…the value of Required Permissions (n) (i.e., RP(n)) at the input to a node n…by means of a set of union operation").

Regarding claim 3, Koved as modified discloses wherein: the one or more assemblies in managed code correspond to an application (Koved: page 3, column 1, third paragraph: "Each Java application class…associated with a set of right, or privileges, granted to the code); and the set of required permissions for each said execution path comprises a union of the permissions for each said execution path (Koved: page 3, column 2; and page 4 column 2: "Since, in general…along paths towards nodes in Nstart. This process associates a set of required requirements RP(n) with each node n in Nstart. More precisely, it computes RP(n) for all n belong to N". "It can be seen that the data flow…by means of a set union operation").

Regarding claim 4, Koved as modified discloses wherein: the assemblies in managed code correspond to a shared library (Koved: page 8, column 1, third paragraph: "For a given application or classes in a library…identify the set of Java 2 Permissions required for each class in the analysis scope"); and the set of required permissions for each said execution path comprises one separate permission set per entry point in the shared library (Koved: on page 1, under ABSTRACT section; and page 2, column 1, under Prior Work section: "This paper presents…compute at each

program point the set of access rights required by the code"…"authorization tests…to the current approach to defining authorization points within code").

Regarding claim 5, this claim has limitations that is similar to those of claims 2 and 3, thus it is rejected with the same rationale applied against claims 2 and 3 above.

Regarding claim 6, Koved as modified discloses wherein one of more of the calls in at least one said execution path is an cross assembly call (Koved: on page 2, column 2, third paragraph: "In the aforementioned works…Java runtime calls one of the SecurityManager authorization methods…to correctly identify authorization requirements").

Regarding claim 7, Koved as modified discloses wherein: the managed code is built to make use of a common language runtime (on page 2, column 2, third paragraph: "In the aforementioned works…Java runtime calls one of the SecurityManager authorization methods…to correctly identify authorization requirements"); each said assembly is packaged as an executable entity or as a data link library entity and each said assembly includes one or more methods (Koved: on page 1, under ABSTRACT section; and page 7, column 2, second and third paragraph: "The tool…to identify the access rights requirements for the product to enable it to run using Java 2 security model").

Regarding claim 8, Koved as modified discloses wherein the simulation of the execution of each said execution path comprises a simulation of the flow of argument data using intra and extra method data flow analysis for each said method (Koved: on page 2, column 1, second and third paragraph; and page 6, column 2, second

paragraph: “To summarize…We present a context sensitive, flow sensitive analysis for computing the access rights requirements of a program.” “To minimize conservativeness…the order of execution of instructions both intra- and inter procedurally thus improving the accuracy of the resulting graph”).

Regarding claim 9, Koved as modified discloses wherein when the executable has permissions to execute that are not less than a union of permission sets for each said execution path, any dynamic execution of the executable will not trigger a security exception (Koved: on page 3, under Authorization Model-Access Rights Invocation Graph section; page 7, column 1, second paragraph; and page 8, column 1, first paragraph: “Performance is improved…NullPointerException in package…”).

Regarding claim 11, Koved as modified discloses a computer readable storage medium having a tangible component including machine readable instructions for implementing the method as defined in Claim 1 (Koved: on page 4, column 1, third paragraph).

Regarding claim 12, this claim has limitations that is similar to those of claims 1-6, thus it is rejected with the same rationale applied against claims 1-6 above.

Regarding claim 13, Koved as modified discloses wherein the managed code environment comprises: a managed code portion including: the assemblies (Koved: see ABSTRACT section); and a virtual machine (Koved: on page 3, column 1, paragraph 3: “Each Java application…the Java Virtual Machine…to the code”); a native code portion including: an execution engine for the virtual machine (Koved: see ABSTRACT section: “Java…protects systems…execute the code…in deployed systems”); and an operating

system under the execution engine (Koved: see ABSTRACT section: "Java…protects systems…execute the code…in deployed systems").

Regarding claim 14, this claim has limitations that is similar to those of claim 7, thus it is rejected with the same rationale applied against claim 7 above.

Regarding claim 15, this claim has limitations that is similar to those of claim 9, thus it is rejected with the same rationale applied against claim 9 above.

Regarding claim 17, Koved as modified discloses wherein the managed code environment enforces partial trust security contexts (Koved: on page 3, column 1, 2 paragraph: "Rather than analyzing…enforce specific security policies…updated to enable the code to execute").

Regarding claim 18, this claim has limitations that is similar to those of claim 11, thus it is rejected with the same rationale applied against claim 11 above.

Regarding claim 19, Koved as modified discloses instructions that, when executed, perform a simulation of the execution of every data and control flow for managed code from which an estimate is derived of the minimum security requirements needed to dynamically execute the managed code without triggering a security exception, wherein the simulation of the execution comprises, for each data and control flow for the managed code, one or more simulated stack walks that include two or more the assemblies (Koved: on page 1, column 1; page 7 under Generation of a Security Policy Decryption section; page 1, column 2, 2nd and 3rd paragraph: " This paper presents…computing the access rights requirements"; page 2, column 2 and last paragraph: "a formalization of stack introspection, which examines authorization based

on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java … at each stack frame, as well as the result of any authorization test encountered" and page 3, column 1 and first paragraph: "the runtime stack … to discover authorization requirements by analyzing all possible paths through the program").

Koved does not explicitly disclose wherein the managed code makes use of a common language runtime (CLR) that is loaded upon the first invocation of a routine, and wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; for each called method: gathering a permission set for the called method; and determining whether the called method calls a subsequent method; and creating a union of the gathered permission sets. However, Scheifler discloses wherein the managed code makes use of a common language runtime (CLR) that is loaded upon the first invocation of a routine, and wherein the simulated stack walk comprises: entering a public entry point of a method in the assembly; gathering a permission set for the method; determining whether the method calls another method; for each called method: gathering a permission set for the called method; and determining whether the called method calls a subsequent method; and creating a union of the gathered permission sets (Scheifler: paragraphs 0016, 0083-0086, 0089, 0091, 0126, 0130 and 0139-0140). Therefore, it would have been obvious to a person skilled art at the time the invention was made to have included in Koved the feature of Scheifler as discussed above in order to increase security by providing

flexible designation of access privileges to code. The systems and methods not only base the access privileges on the source of the code (i.e., whether the code is trusted or untrusted), but also on the identity of the principal on whose behalf the code is being executed (i.e., whether the principal requesting the code execution is trusted or untrusted) (Scheifler: paragraph 0026).

Regarding claim 20, this claim has limitations that is similar to those of claim 7 and 14, thus it is rejected with the same rationale applied against claims 7 and 14 above.

Regarding claim 21, this claim has limitations that is similar to those of claim 13, thus it is rejected with the same rationale applied against claim 13 above.

Regarding claim 22, this claim has limitations that is similar to those of claim 20, thus it is rejected with the same rationale applied against claim 20 above.

Regarding claim 24, this claim has limitations that is similar to those of claim 16, thus it is rejected with the same rationale applied against claim 16 above.

Regarding claim 25, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 26, Koved discloses means for processing; means for storing information in memory coupled to the means for processing; virtual machine means, stored in the memory, in a managed code portion, for operating a plurality of assemblies in managed code, wherein the managed code is a managed shared library or an executable and is in the managed code portion; execution engine means, in a native code portion, for executing the virtual machine means (Koved: on page 3, column 1,

third paragraph: “Each Java application class…Java Virtual Machine…privileges, granted to the code”); means, in the native code portion, for providing an operating system (Koved: on page 3, column 1, third paragraph; and page 4, column 1, first paragraph: “Each Java application class…Java Virtual Machine…privileges, granted to the code”); means for making a call in the managed code portion for access by one assembly to another assembly for which a permissions set is required (Koved: see ABSTRACT section on page 1); means in the managed code portion for gathering the permissions set from each said call (Koved: on page 4, column 1, second paragraph); means for deriving a union of the gathered permissions sets (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section); and means in the managed code portion for simulating the execution of all possible execution paths for the managed shared library or the executable to derive therefrom the derived union of the gathered permissions sets wherein the mans for simulating the execution performs, for each execution path, one or more simulated stack walks that each include a plurality of assemblies (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section; page 2, column 2 and last paragraph: “a formalization of stack introspection, which examines authorization based on the principals (signers and/or origin of the code) currently active in a Thread stack at run-time, as is found in Java … at each stack frame, as well as the result of any authorization test encountered” and page 3, column 1 and first paragraph: “the runtime stack … to discover authorization requirements by analyzing all possible paths through the program”).

Koved does not explicitly disclose wherein the one or more simulated stack walks comprise: means for entering a public entry point of a method in the assembly; means for gathering a permission set for the method; means for determining whether the method calls another method: for each called method: means for gathering a permission set for the called method: means for determining whether the called method calls a subsequent method; and means for repeating the previous gathering and determining until any gathered permission set is duplicative: and means for creating a union of the gathered permission sets. However, Scheifler discloses wherein the one or more simulated stack walks comprise: means for entering a public entry point of a method in the assembly; means for gathering a permission set for the method; means for determining whether the method calls another method: for each called method: means for gathering a permission set for the called method: means for determining whether the called method calls a subsequent method; and means for repeating the previous gathering and determining until any gathered permission set is duplicative: and means for creating a union of the gathered permission sets (Scheifler: paragraphs 0016, 0083-0086, 0089, 0091, 0126, 0130 and 0139-0140). Therefore, it would have been obvious to a person skilled art at the time the invention was made to have included in Koved the feature of Scheifler as discussed above in order to increase security by providing flexible designation of access privileges to code. The systems and methods not only base the access privileges on the source of the code (i.e., whether the code is trusted or untrusted), but also on the identity of the principal on whose behalf the code is being

executed (i.e., whether the principal requesting the code execution is trusted or untrusted) (Scheifler: paragraph 0026).

Regarding claim 27, Koved as modified discloses means for compiling the assemblies from an intermediate language code and metadata into native code; and means for loading the native code with a Common Language Runtime loader in the native code portion to load the compiled native code, wherein the execution engine means executes the compiled native code in the native code portion (Koved: on page 3, column 1, first paragraph: "Permission.implies…test cases").

Regarding claim 28, Koved as modified discloses wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the assemblies (Koved: on page 3, column 1, third paragraph: "Each Java application class…Java Virtual Machine…privileges, granted to the code").

Regarding claim 29, Koved as modified discloses wherein the execution engine means in the native code portion further comprises a compiler to compile each said assembly into native code for execution by the native code portion (Koved: on page 3, column 1, third paragraph: "Each Java application class…Java Virtual Machine…privileges, granted to the code").

Regarding claim 30, Koved as modified discloses wherein the execution engine means in the native code portion further comprises: a Just In Time compiler to compile each said assembly into native code; and a common language runtime loader to load

the compiled native code for execution by the native code portion (on page 3, column 1, third paragraph: "Each Java application class…Java Virtual Machine…privileges, granted to the code").

Regarding claim 31, Koved as modified discloses means, in the native code portion, for forming a response to the call; and means for returning the response to the first assembly in the managed code portion (Koved: on page 3, column 1, third paragraph; and page 4, column 1, first paragaraph: "Each Java application class…Java Virtual Machine…privileges, granted to the code").

Regarding claim 32, Koved as modified discloses wherein: the managed code is built to make use of a common language runtime; each said assembly is packaged as an executable entity or as a data link library entity; and each said assembly includes one or more methods (Koved: on page 1, under ABSTRACT section; and page 7, column 2, second and third paragraph: "The tool…to identify the access rights requirements for the product to enable it to run using Java 2 security model").

Regarding claim 33, Koved as modified discloses wherein the simulation of the execution comprises, for each said execution path, a simulation of the flow of argument data using intra and extra data flow analysis for each said method (Koved: on page 2, column 1, second and third paragraph; and page 6, column 2, second paragraph: "To summarize…We present a context sensitive, flow sensitive analysis for computing the access rights requirements of a program." "To minimize conservativeness…the order of execution of instructions both intra- and inter procedurally thus improving the accuracy of the resulting graph").

Regarding claim 34, this claim has limitations that is similar to those of claim 9, thus it is rejected with the same rationale applied against claim 9 above.

Regarding claim 36, Koved as modified discloses wherein each call in each simulated stack walk has a corresponding permissions set (Koved: on page 3 under Authorization Model-Access Rights Invocation Graph section: “For any node…set of required Permissions for n”).

Regarding claim 37, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 38, this claim has limitations that is similar to those of claim 26, thus it is rejected with the same rationale applied against claim 26 above.

Regarding claim 39, this claim has limitations that is similar to those of claims 27 and 28, thus it is rejected with the same rationale applied against claims 27 and 28 above.

Regarding claim 40, this claim has limitations that is similar to those of claim 30, thus it is rejected with the same rationale applied against claim 30 above.

Regarding claim 41, this claim has limitations that is similar to those of claim 22, thus it is rejected with the same rationale applied against claim 22 above.

Regarding claim 42, this claim has limitations that is similar to those of claim 8, thus it is rejected with the same rationale applied against claim 8 above.

Regarding claim 43, this claim has limitations that is similar to those of claims 9 and 15, thus it is rejected with the same rationale applied against claims 9 and 15 above.

Regarding claim 45, this claim has limitations that is similar to those of claim 17, thus it is rejected with the same rationale applied against claim 17 above.

Regarding claim 51, Koved as modified discloses wherein the union of the permission sets separately identifies a permission set for each public entry point of the library (Koved: on page 2, column 2, third paragraph; page 3, column 1, first paragraph; and page 3 and page 4, under Authorization Model section: "In this paper…an invocation graph and data flow analysis…more accurate authorization information." "Our approach…discover authorization requirements by analyzing all possible paths through the program." "It can be seen…the value of Required Permissions (n) (i.e., RP(n)) at the input to a node n…by means of a set of union operation").

Conclusion

20. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Kershenbaum et al. discloses method and apparatus for automatically determining optimum placement of privileged code locations in existing code (US 2004/0040017).

- Stefik et al. discloses method and apparatus for executing code in accordance with usage rights (US 2003/0225698).
- Anand et al. discloses flexible and dynamic derivation of permissions (US 6044466).
- Apperson et al. discloses system and method for safety distributing executable objects (US 5978484).
- Lamacchia et al. discloses applying a permission grant set to a call stack during runtime (US 7076557).
- Scheifler et al. discloses stack-based access control using code and execution identifiers (US 6138238).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to TRANG DOAN whose telephone number is (571)272-0740. The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on (571) 272-3795. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Trang Doan/
Examiner, Art Unit 2431
/Syed Zia/
Primary Examiner, Art Unit 2431